

IPR2024-00838

US Patent No. 10,901,191

EXHIBIT 2003

“Chapter XXXIII - 3-D Visualization on Mobile Devices”

Sanna and Lamberti (Copyright 2008)

**from Handbook of Research on User Interface Design and Evaluation for
Mobile Technology, 3-D Visualization on Mobile Devices, 2008, DOI:
10.4018/978/1/59904-871-0.ch033**

Source

Exhibit 6 to Opening Claim Construction Brief

Lumenco, LLC v. Giesecke+Devrient GmbH (E.D. Tex. No. 2:23-cv-00100-JRG)

Dkt. No. 93-6 – April 2, 2024

EXHIBIT 6



www.igi-global.com/ondemand



InfoSci[®]-onDemand Chapter Download



Purchase individual research articles, book chapters, and teaching cases from IGI Global's entire selection.

www.igi-global.com/ondemand

This publication is protected by copyright law of the United States of America codified in Title 17 of the U.S. Code, which is party to both the Universal Copyright Convention and the Berne Copyright Convention. The entire content is copyrighted by IGI Global. All rights reserved. No part of this publication may be reproduced, posted online, stored, translated or distributed in any form or by any means without written permission from the publisher.

Chapter XXXIII

3-D Visualization on Mobile Devices

Andrea Sanna

Politecnico di Torino, Italy

Fabrizio Lamberti

Politecnico di Torino, Italy

ABSTRACT

Recent improvements in technology of last generation mobile devices (smartphones, Personal Digital Assistants, Ultra-Mobile PCs) have opened up challenging new scenarios in 3-D ubiquitous visualization. Entertainment, cultural heritage, telemedicine, and distance learning are only a few of the disciplines that can take advantage of 3-D model visualization on mobile and hand-held devices. This chapter investigates the main methodologies used to display and navigate complex scenes and heavy datasets on mobile devices. Both local and remote rendering software techniques are considered with respect to solutions based on hardware acceleration. Moreover, issues related to the design of suitable graphics user interfaces are tackled. Finally, a solution for high performance visualization based on the remote rendering approach and enabling interactive manipulation of 3-D scenes composed by millions of polygons at 30 frames per second even on 50 Kbit/s wireless communication channels is presented.

INTRODUCTION

The advent of large-bandwidth wireless networks and improvements in graphics hardware have opened up exciting and intriguing new scenarios concerning ubiquitous visualization on mobile devices. Mobile phones and personal digital assistants (PDAs) are now able to display 3-D

graphics contents and the demand for visualization applications is rapidly increasing.

In the past years, reduced hardware resources, limited bandwidth communication channels and inadequate graphics users interfaces (GUIs) strongly limited the use of mobile devices for 3-D graphics. Nowadays, mobile phones (often named smartphones), PDAs, and tablet PCs are able to

3-D Visualization on Mobile Devices

connect to the Internet via high-speed networks (like for example IEEE 802.11 and UMTS) and high-performance CPUs, able to provide a support for multimedia applications, are becoming available. Several solutions integrating a CPU and a GPU (graphics processing unit) have also been developed (it is worth observing that a spread diffusion of graphics applications for desktop PCs began when accelerated graphics adapters appeared on the market). Moreover, hand-held devices can be equipped with a wide range of accessories allowing researchers to design and implement a complete and effective set of multimodal interfaces.

All these technological advances make ubiquitous graphics a new and challenging field of research. Today mobile devices are able to locally manage and display several thousands of polygons per second, as well as play high-resolution video streams in real time. Additionally, frameworks for managing mobile interactive visualization of highly complex datasets composed of millions of polygons/voxels have been developed. This means that ubiquitous 3-D visualization can now be performed in an efficient and effective way, thus helping scientists and doctors to achieve a clearer understanding of the nature of data under investigation, and supporting the entertainment market. Many fields of research and applications can take advantage of ubiquitous 3-D visualization: education, art, advertising, medicine, entertainment, and so on.

In order to provide an overall view on the state of the art of 3-D visualization on mobile devices, the chapter starts following a scheme based on the device workload. Thus, *local visualization* is considered first in the following Section. In this technique, the entire visualization process (geometry management, rendering, and display) is performed exploiting exclusively the computational resources of a mobile device often named fat client. An entire Section is then devoted to *remote visualization*. Here, a remote server is in charge of processing the geometry and rendering the scene. The mobile device, also called thin client, only has to display a set of still images or a video stream. Remarks concerning the design and the development of mobile GUIs for 3-D are then

presented. Finally, the design and experimental results of a framework for high-performance 3-D visualization on mobile devices based on the remote rendering approach are presented.

LOCAL COMPUTATION

The term local computation refers to a situation where the entire geometry of the model can be stored on the mobile device and locally available computational resources are sufficient to render and display the scene (Zhou, 2006). The adjective sufficient can take different meanings according to the application fields being considered: a few frames per second can be considered sufficient when a limited interaction is required, but 30 frames per second, or more, can be necessary for video games or other “real-time” applications.

It is worth stressing that local computation requires data to be stored in the memory of the mobile device; beyond implications related to hardware resources, a problem of security can arise. Sometimes, it is preferred not to move visualization datasets outside “secure” environments, because data can contain reserved and/or highly confidential information. On the other hand, limited network resources are not involved in the visualization process, at least for single-user applications. Thus, visualization performances do not depend on network coverage and bandwidth.

From the software point of view, several solutions have been proposed to design and implement 3-D rendering engines for mobile devices. MiniGL (2006) and PocketGL (2006) were two of the first examples; they consist of a 3-D toolkit exporting a large number of utility functions needed to manipulate 3-D objects. They exhibit several similarities with OpenGL (Woo, Neider, Davis, & Shreiner, 1999). In fact, OpenGL code and tutorials can be used as a starting point to implement MiniGL- and PocketGL-based applications.

Another set of APIs named OpenGL ES, similar to OpenGL but specifically tailored for mobile devices, has been recently proposed (OpenGL ES, 2006). It is worth observing that the acronym ES means embedded system in order to outline

3-D Visualization on Mobile Devices

the vocation of this library. Similarly to OpenGL, OpenGL ES is a set of low-level functions representing a subset of OpenGL calls capable of taking advantage of advanced embedded graphics. Several implementations are already available like Vincent (2006) and Hybrid's Gerbera (2006). Another open-source library similar to OpenGL is Klimt (2006), formerly known as SoftGL; Klimt is targeted to hardware independence and is available for many mobile platforms.

As Direct3D (Glidden, 1997) is a competitor of OpenGL, Direct3D Mobile (Microsoft, 2006) is likely to become a competitor of OpenGL ES. By using Direct3D Mobile, developers can take advantage of their existing desktop computer Direct3D skills and code to produce rich three-dimensional gaming environments for Windows Mobile devices. Direct3D Mobile takes full advantage of graphics hardware support, and enables the development of high-performance three-dimensional rendering on mobile devices.

Several sets of APIs for J2ME (Java 2 Platform, Micro Edition) have also been proposed. M3D (2006) is able to scale for both limited resource devices (like, for example, CPUs without support for floating point computation) and high-end devices equipped with specialized graphics hardware. M3G - JSR 184 (2006) is built as a wrapper on top of the Hybrid OpenGL ES API implementation; it is a J2ME optional package that allows moderate complexity 3-D graphics to be rendered at interactive frame rates on mobile devices.

Three-dimensional scenes can also be inspected on mobile devices by means of VRML (virtual reality modeling language) browsers. Pocket Cortona (Parallel Graphics, 2006) was one of the first VRML browsers for viewing VRML scenes on wireless devices; it is worth remarking that the latest version available can take advantage of the Xscale technology (Intel Xscale, 2006) to improve performances and reduce power consumption over a wide spectrum of hand-held and mobile devices.

Unfortunately, all the solutions mentioned are able to produce a smooth interactive visualization only for relatively simple scenes. For instance, Nokia 6630 smartphone is able to render about

60k triangles per second; if a frame rate of 30fps is required, only simple scenes modeled by a few thousands of polygons can be managed. Although the screen resolution of a hand-held device cannot always allow the user to appreciate highly detailed geometries, complex 3-D model visualizations can be required when displays provide a relatively large visualization area (as, for example, with last generation PDAs and existing Tablet PCs). In this case, the rendering engine has to be able to simplify the scene geometry to allow an interactive visualization.

Proposed techniques that modify the geometry (reducing the number of polygons without affecting in a noticeable way the appearance of the model) are based on level of detail (LOD) algorithms; the basic idea is to display a simplified version of the geometry when the scene is navigated, while the maximum level of detail is only displayed for fixed points of view. LOD algorithms can be classified as static or dynamic; a static LOD algorithm performs an off-line simplification of the geometry (the computational cost of simplification is not paid during the rendering) by producing a set of new models representing different approximations of the original objects. During visualization, the rendering engine chooses the appropriate representation for each object according to the distance from the observer (far objects are represented by less-detailed geometries). Static LOD algorithms are able to reduce computational costs; however, more memory space is needed to store the different representations. On the other hand, dynamic solutions are able to compute/simplify online the geometry of the scene. Dynamic strategies are able to produce smooth (almost unnoticeable) transitions between visualizations of different levels of detail, but they result in large computational costs often not affordable on mobile devices. Exhaustive surveys of generic multiresolution modeling techniques have been provided by Garland (1999) and Luebke (2001).

An example of LOD algorithm implemented for mobile devices has been developed by Zunino (Zunino, Lamberti, & Sanna, 2003). In this work, a continuous multiresolution modeling technique has been used to achieve a satisfactory trade-off

3-D Visualization on Mobile Devices

between quality and interactivity. The proposed methodology allows the user to set a desired frame rate value that is used as an objective function. The scene complexity is scaled down when the current frame rate is lower than the threshold value, while it is increased if the frame rate trend is positive. Geometry decimations/refinements are performed according to a cost function able to take into account both the distance of the objects from the observer and the impact of topological changes on the visualization.

Another approach that can be used to reduce the computational cost on hand-held and mobile devices is based on image-based rendering (IBR) algorithms. IBR techniques use a limited set of key frames (considered as input images) to “interpolate” all the intermediate pictures (McMillan, 1999). An example of application of IBR algorithms for mobile devices has been proposed by Chang and Ger (2002). The designed framework consists of two parts: a model constructor and an interactive warper. The job of the model constructor is to convert 3-D models into depth images that are amicable to 3-D warping. The model constructor can be considered as a preprocessing step; therefore, it may run remotely on desktop computers rather than on mobile devices. The 3-D warper runs on mobile devices; it accepts user inputs and displays the new views interactively. An implementation of the 3-D warping algorithm presented above has been proposed by McMillan and Bishop (1995). Unfortunately, IBR techniques often produce artifacts due to the camera position; for this reason, they represent a feasible solution only in a limited number of visualization scenarios.

Figure 1 shows a typical local visualization scenario: the client stores in its local memory the 3-D model to be visualized (the scene can be downloaded from a remote server) and a local application (for instance, a VRML browser or a viewer based on OpenGL ES technology) uses CPU and GPU (if available) of the mobile device in order to process the graphics primitives and render the scene.

REMOTE COMPUTATION

Although local computation strategies can constitute a viable solution to visualize “relatively simple” scenes, geometries described by millions of textured polygons or large-volume datasets are beyond the capability of any mobile device.

A completely different approach for dealing with these geometries consists in dividing rendering and visualization workloads between a remote server and the mobile device. According to this philosophy, the only purpose of the mobile device is to display a flow of data (a sequence of still images or a video stream) coming from a remote server that is in charge of storing and managing the geometry, rendering the scene and sending resulting frames to the client (Zhou, 2006). The remote server can be a graphics workstation as well as a cluster of PCs or any other specialized hardware able to manage and render extremely complex models. It is worth remarking that in remote visualization strategies, only data representations are delivered to the mobile device that sends back user commands to inspect the scene, while data can be kept on the secure remote server. This solution

Figure 1. Local visualization



3-D Visualization on Mobile Devices

can require large bandwidth connections if high resolution streams have to be delivered; moreover, the client should provide a support for efficiently carrying out multimedia operations in order to perform on-the-fly processing of incoming video streams. A general classification of visualization algorithms based on a client-server approach has been provided by Martin (2000).

In remote computation, two different approaches are possible: computing a continuous set of still images or generating a video stream. A video, such as an MPEG stream, is able to take advantage of both compression of the single frame and temporal redundancy in a sequence of consecutive frames. On the other hand, the visualization of a video stream requires more computational resources than a single picture; this can be a problem for certain classes of hand-held and mobile devices not provided with powerful multimedia capabilities.

A simple but effective idea for addressing these issues was proposed by Stegmaier et al. (Stegmaier, Magallón, & Ertl, 2002); an X-server is remotely controlled by a PocketPC version of VNC - virtual network computing (Richardson, Stafford-Fraser, Wood, & Hopper, 1998). This approach is extremely attractive since it is application-independent and does not require any change in the interface. On the other hand, two main drawbacks can be identified. The transmission of the entire screen of the remote server can involve a large network load; hence, it could be inadequate for limited bandwidth channels. Furthermore, the advantage of being application-independent could constitute a drawback since the system cannot be optimized for any specific application.

In order to reduce network requirements, VirtualGL (2006) was proposed. The idea is to send a compressed picture to the mobile client instead of the raw screen of the remote console; different coding schemes have been evaluated by Stegmaier et al. (Stegmaier, Diepstraten, Weiler, & Ertl, 2003). Although a compressed frame requires less bandwidth than a raw one, computation resources needed at the client side to decode and display a compressed picture increase. Lamberti et al. (Lamberti, Zunino, Sanna, Fiume, & Maniezzo,

2003) showed that the frame rate obtained using a Compaq iPaq H3630 PDA is maximum when raw frames are sent to the client, since the mobile device is not able to efficiently decode and display compressed images.

The solution proposed by Lamberti et al. (2003) adopts a specialized remote server composed by a cluster of PCs, where the rendering is piloted by a distributed framework in order to take advantage of multiple hardware accelerated graphics adapters. An OpenGL application runs over each PC of the cluster, and the computation workload is split among all the available GPUs. A master PC is in charge of receiving contributions from the other units, reassembling the resulting frame and sending the picture to the mobile client. The authors demonstrated that this approach allows a visualization frame rate of 7 fps to be achieved (at a resolution of 120×120 pixels) that is independent of the scene complexity (in fact, the computational cost is paid at the remote server side).

A high performance grid, based on Web services, was also proposed by Grimstead et al. (Grimstead, Avis, & Walker, 2005) in order to allow mobile clients to display large medical datasets; the performance of this solution is greatly affected by the grid effectiveness, which actually depends on network connections. While a PC cluster is typically obtained by using high speed (Gigabit or fast-Gigabit Ethernet) switches, grid connections often suffer from high latency times, and are not always suited to real-time/interactive graphics applications.

A platform-independent framework written in Java was proposed by Engel et al. (Engel, Sommer, & Ertl, 2000); a similar solution, tailored for low-bandwidth systems, was presented by Beerman (2003).

Some commercial solutions have been also presented; Silicon Graphics developed Vizserver (Silicon Graphics, 2005), which is able to provide application-transparent remote access to high-end graphics resources for multiple simultaneous users. Similarly, IBM developed a product called Deep Computing Visualization (IBM, 2006).

Among the commercially available solutions, it is also worth recalling the recent RealityServer by

3-D Visualization on Mobile Devices

Figure 2. Remote rendering

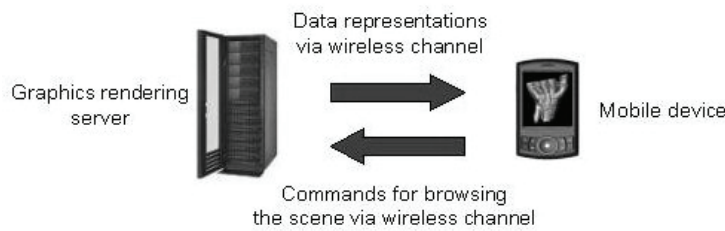
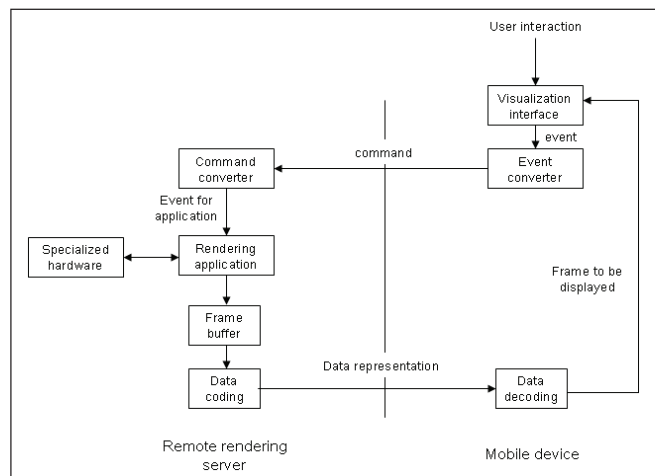


Figure 3. Sequence of operations in a remote rendering session



mental images (2006). RealityServer is a server-based platform allowing 3-D contents developed using CAD/CAM packages, to be navigated by remote client devices equipped with a Web browser that supports Flash or DHTML and JavaScript. RealityServer supports both local and remote rendering. When local rendering is used, a Web browser plug-in named RealityPlayer receives and manages the rendering of automatically-created reduced representations of the original 3-D content generated by the RealityServer. The system can additionally be deployed so that the server renders a photorealistic version of the original model, and sends it to the client for display after scene manipulation. When remote rendering is used, only a sequence or encoded stream of rendered

images is delivered to front-end client devices. This allows in principle integration with any kind of client device. However, even if rendering and streaming can be processed in real time in certain rendering modes, bandwidth restrictions may adversely affect interactivity. In fact, according to *mental images*, white paper (2005), the visualization of a 3-D content on a smartphone/PDA at a resolution of 320×240 at 10 frames per second requires a constant bandwidth around 360 Kbit/s that largely exceeds even the average data rate of UMTS. Moreover, a 720 Kbit/s throughput is needed to handle scene manipulations at 20 frames per second.

Figure 2 shows a typical scenario where visualization on a mobile device is performed through

3-D Visualization on Mobile Devices

a remote rendering technique. A remote rendering server computes the scene for the mobile client, and only a 2-D representation is sent over the network (as a flow of still images or as a video stream). The user application at the client side has to be designed to display data coming from the server and to capture commands issued to manipulate the scene. Commands have to be sent to the server to be processed; the rendering server receives commands from the client and updates the rendering of the scene accordingly. Figure 3 shows the sequence of operations of a typical remote rendering session in further detail.

GUI DESIGN REMARKS

The development of 3-D visualization environments on mobile devices requires not only performance constraints, but also user interaction considerations to be taken into account. In fact, 3-D navigation, and particularly 3-D manipulation, require complex GUIs to be developed. It is worth remarking that when a mobile visualization application only relies on local rendering, there is no need to completely replicate interfaces available in workstation-based scenarios. Thus, GUIs can be designed from scratch, taking into account the suggestions of research in the field on human-computer interaction (HCI). Nevertheless, when considering remote visualization solutions possibly involving collaborative sessions, different considerations arise. This is due to the fact that the remote server runs a visualization application characterized by its own GUI and interactions available at the workstation-based visualization site have to be made available also on a mobile device that is often characterized by a display limited both in size and resolution, a constrained keyboard, and a navigation button or stylus only partially emulating the behavior of a mouse. A communication channel is established between the remote server and the mobile device to report events generated by the user to the rendering server where they are translated into a modification of the visualization parameters. Below, some of the possibilities investigated in order to

achieve effective visualization interaction will be reviewed. Their advantages and drawbacks will be also outlined.

In Grimstead et al.'s work (2005) complex 3-D datasets are represented through a scene graph hosting both polygons and voxels. While interacting with the dataset, the user can select an object by tapping on the display of the mobile device. A mapping algorithm allows the coordinates of the selection on the client device to be translated into a precise point in the remote scene graph. Depending on the specific manipulation context, it is possible to roll camera, move objects, and rotate around objects. The mobile client requests, from the rendering server, the set of available interactions for the particular object through a dedicated interface based on Web services. Available interactions are presented to the user through several menu options. A collaborative modality allows multiple users to concurrently manipulate a common scene. In this modality, each user is represented by an avatar linked to the camera, that is, to the user point of view. During a collaborative session, the user can follow the avatar of another user, thus sharing the same point of view. Additionally, depending on the security configuration being adopted, a user can control the camera associated with other users thus, making a sort of remote control of other devices possible. A possible application for this feature could be the use of a PDA to control a large-scale display. The authors expect to integrate more advanced collaboration functionalities, including the possibility of advertising points of view of common interest, the feasibility of linking to other cameras in order to implement guided tours within the virtual environment, and the adoption of audio and audio/gestural communications to improve cooperation. In the situation presented, a GUI is specifically designed for the application running on the mobile device. The only limitation to the complexity of the interface is related to the capabilities of the development environment available for the selected platform. The critical role is played by the mapping strategy that acts as a wrapper between the 2-D coordinate system of the mobile device display and the 3-D environment represented by the scene graph.

3-D Visualization on Mobile Devices

In Engel et al.'s work (Engel, Ertl, Hastreiter, Tomandl, & Eberhardt, 2000), a hybrid approach, based on both local and remote visualization, is adopted to support high-quality manipulation of volume data. A client application designed for mobile devices provides the user with a slicing tool deployed in Java/Java2D, permitting inspection of a complex medical dataset obtained from CT scans. Users can select a region of interest, and a low-quality 3-D rendering can be obtained on the device using Java3D. If a high-quality visualization is required, volume data can be transferred to a remote rendering server that actually performs the rendering, compresses the resulting frame, and transmits the 2-D picture ready for display to the mobile device. The local GUI allows the user to apply various transformations on the dataset including rotations, translations, and zooms. It is possible to change the transfer function, define orthogonal clipping planes, and perform iso-surface reconstruction. In this case, the GUI available at the remote-rendering server, based on Open Inventor, has been completely reconstructed on the mobile device using Java graphics functionalities. In this way, standard manipulators of Open Inventor are made available on the client side. Events generated by mouse, key presses, and widgets, including thumb wheels, sliders, push buttons, and context menus, are directly mapped onto the corresponding instances in the remote environment. The client has been tested on portable PCs, but it can be potentially migrated to Java-based hand-held devices.

Stegmaier et al. (2002) deployed VNC on a PocketPC device to remotely control a visualization application running on a rendering workstation. No particular solution is adopted to deal with the GUI on the mobile device. Thus, complex GUIs possibly available at the remote workstation are interleaved with visualization data and presented on the mobile device display. However, they are difficult to manage on the limited size display of a PDA device.

Woodward et al. (Woodward, Valli, Honkamaa, & Hakkarainen, 2002) presents a solution that enables engineers and other users to view and interact with 3-D CAD files using a wireless

phone connection and a PDA device. It applies live video compression to transfer the continuously changing CAD view at the server workstation to the client PDA device. The client GUI lets the user rotate, pan, and zoom into the CAD model, as well as ask for distances, angles, and radii between points in the model. The author claims that other CAD interaction commands are straightforward to implement.

Burigat and Chittaro (2005) developed a dedicated application allowing mobile users to navigate VRML scenes by exploiting GPS-based localization information. Both local and remote rendering can be exploited to deal with scene visualization, depending on dataset complexity. In this case, no visualization console exists. There is no GUI to be replicated on the mobile device, and development efforts have been concentrated on the optimization of visualization performance. Thus, a simplified GUI has been designed to let the user perform manual or GPS-assisted navigation. The GUI provides a 3-D visualization area, status information, object details, and widgets to control the point of view when in manual mode.

Finally, a preconfigured solution is exploited by *mental images* (2006). In their RealityServer framework, imported 3-D contents are static, and existing graphics applications, possibly designed to manipulate the scene, cannot be reused. To allow interaction and customization, developers must write new application logic using a JavaScript-based API. The main drawback of this solution is that interaction logic is stored at the remote side together with 3-D data, and only limited customizations are possible at the mobile side.

HIGH-PERFORMANCE MOBILE 3-D REMOTE VISUALIZATION

Next, we present a complete framework for remote visualization in mobile environments based on distributed rendering and video streaming techniques. The proposed architecture can be transparently applied to professional visualization scenarios thus, enabling multiple mobile users to remotely control a variety of graphics applications

3-D Visualization on Mobile Devices

based on OpenGL libraries. Experimental results showed that a 3-D scene, composed by more than three-million textured polygons can be visualized over wireless communication networks at about 30 fps at a resolution of 512×512 pixels on mobile and hand-held devices.

Remote Rendering Subsystem

The rendering subsystem at the server-side relies upon a software module named *distributed visualization service (DVS)* that is in charge of managing a remote rendering application deployed in a cluster-based distributed environment and communicating with a visualization application running on the mobile device. Through the *DVS*, multiple mobile users can control an OpenGL application running on a high-performance rendering facility during collaborative and interactive visualization sessions. It is worth observing that the *DVS* is also responsible for managing a scheduling subsystem permitting handling of concurrent manipulation commands that can be issued by both a visualization console at the server side and by simultaneous mobile users participating in the session.

The internal behavior of the *DVS* can be summarized as follows. Mobile clients connect to the *DVS* and start or join a visualization session by specifying several parameters including resolution, bitrate, and quality. Scene manipulation commands issued by mobile users are translated into a suitable format for transmission to the *DVS* over a wireless link. *DVS* processes incoming data, and extracts relevant information to be passed to the OpenGL callback layer for adjusting mapping and rendering configurations. Then, local graphics resources are used to execute OpenGL directives, and to generate the requested 2-D rendering frame to be sent to the mobile device for visualization. Graphics resources at the server side rely upon a Chromium (Humphreys, Houston, Ng, Frank, Ahern, Kirchner, & Klosowki, 2002) based architecture in order to distribute the execution of OpenGL rendering commands over a cluster of PCs. Chromium provides a software framework to unify the rendering power of a set of graphics

accelerators distributed in cluster nodes through a virtualized interface based on OpenGL API. In a common Chromium architecture, a client submits OpenGL commands to one or more graphics servers. Each server is in charge of rendering only a part of the output image, which is treated as a tiled display and is later reassembled for visualization. In this way, even extremely complex scenes can be interactively rendered assuming that a sufficient number of graphics server nodes is available.

Chromium framework is based on a client-server paradigm relying on basic modules named *stream processing units (SPU)*. Specific SPUs are responsible for subdividing 3-D context into tiles and for distributing graphics workload to server nodes (*Tilesort SPU*), carrying out the rendering (*Readback SPU*), sending back rendering parts to the client node (*Send SPU*) and finally, reassembling the overall image (*Render SPU*). In order to extend Chromium capabilities and enable it to support remote visualization, existing SPUs have been modified and new SPUs have been developed. In particular, a *Video SPU* has been developed to extract a frame-based video sequence from the assembled rendering framebuffer, and to serially distribute multiple copies of this sequence to ad-hoc components that are in charge of carrying out MPEG encoding and managing streaming to remote visualization clients according to the parameters passed during session initialization.

In particular, encoding is performed by a software module, called *Encoding Service*, that receives a frame-based raw video sequence and scales it to the requested resolution, converts rescaled frames to the YUV format suitable for MPEG processing, performs MPEG encoding generating an MPEG ES (Elementary Stream) bitstream (MPEG1, 1992), and finally passes encoded video to a streaming component. The *Encoding Service* module has been developed as a separate application that can be deployed on a dedicated machine communicating with the *Video SPU* over a TCP connection. In this way, the overhead on the client node is reduced and a performance speedup can be achieved. Beside that, multiple instances of the *Encoding Service* can be allocated (even on separate nodes), allowing for

3-D Visualization on Mobile Devices

an effective management of heterogeneous client devices requesting video streams characterized by a different resolution, bitrate, and/or quality. The additional benefit of such a modular architecture is that the *Video SPU* and the *Encoding Service* can work in parallel. Thus, when a frame is being encoded, the *Video SPU* can carry out the next rendering cycle. According to this pipelined approach, when computing the frame rate that can be achieved in the video stream generation process, only the duration of the longest phase has to be taken into account.

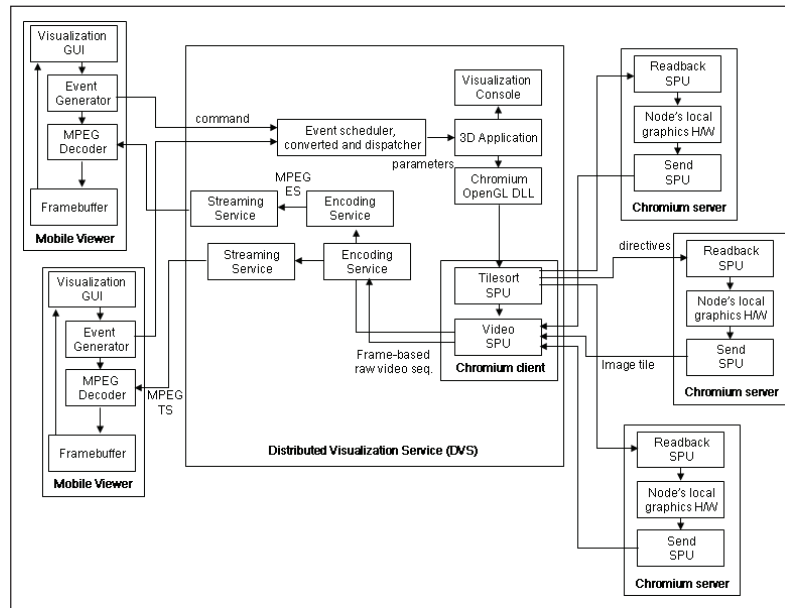
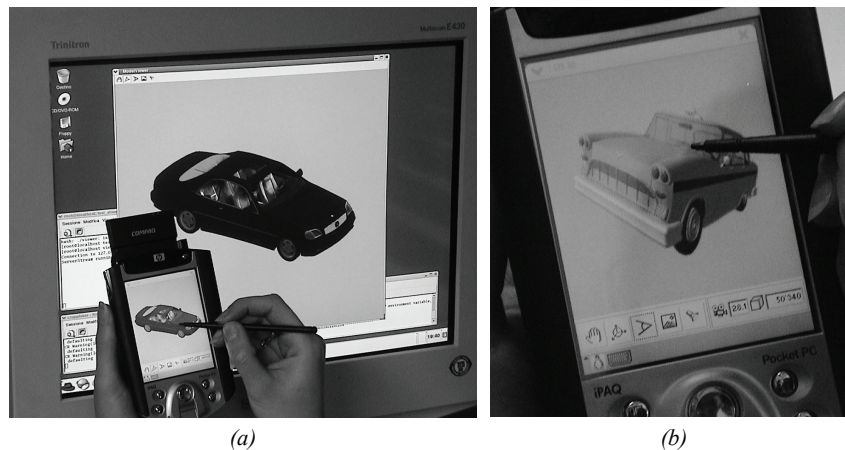
MPEG frames generated by the *Encoding Service* are passed to the *Streaming Service* for distribution to mobile visualization devices using MPEG TS (Transport Stream). The MPEG TS format, defined in MPEG-2 specifications (MPEG2, 1994), has been chosen since it provides effective support for interactive transmission and visualization of multimedia contents over unreliable networks. Essential timing information is passed by means of time stamps inserted into each video data unit header. In particular, periodical transmissions of PCR (program clock reference) time stamps enable continuous synchronization of the receiver and transmitter clocks. PTS (presentation time stamp) indicates the time at which an encoded frame should be removed from the receiver buffer, instantaneously decoded and presented for display, while DTS (decoding time stamp) indicates the time at which an encoded frame should be instantaneously removed from the receiver buffer and decoded. Thus, streamed data incorporate sufficient information to carry out synchronization between encoder and decoder, allowing for the correct handling of out-of-order or delayed, and possibly corrupted, data delivery. In this way, data can be streamed directly over a UDP connection without introducing the overhead of additional protocols. Nevertheless, in the future, the effects of the introduction of alternative synchronization and control protocols (i.e., RTP, RTCP, RTSP) are expected to be evaluated.

The structure of the *Streaming Service* internal logic can be summarized as follows. The MPEG ES data flow received by the *Encoding Service* is first converted into an MPEG PES (Packetized Elementary Stream) (MPEG1, 1992) video se-

quence. This operation allows for the generation of variable length packets embedded with DTS and PTS information. MPEG PES stream is then converted into MPEG TS fixed-length data units including PCR information, which are placed in a FIFO output buffer for transmission. MPEG TS packets are extracted from the buffer according to the transmission scheme, and placed in a multicast UDP socket, thus, implementing simultaneous streaming to multiple mobile clients. The overall layout of the proposed architecture is illustrated in Figure 4.

Mobile Visualization Client

In the designed architecture, mobile devices equipped with a dedicated software application, called *Mobile Viewer*, simply display virtual 3-D scenes that are actually produced at a remote rendering facility. *Mobile Viewer* handles manipulation commands that are issued by mobile users through common input devices like touch screens, directional pads, and/or customizable buttons. *Mobile Viewer* application has been deployed using Gtk+ 1.2 and X11 on both an HP iPaq H5500 PDA running Familiar Linux 0.6.1 and on a Compaq T1000 Tablet PC running Linux RedHat 8.0. A porting of the source code also permitted evaluation of application performances in a Microsoft Windows XP-based native installation on the selected Tablet PC. The architecture of the client application can be summarized as follows (see Figure 4). A *Visualization GUI* component is in charge of handling events generated by the interface and passing them to an *Event Generator* module that converts them into a format suitable for transmission. The *Visualization GUI* module is also responsible for supervising the render area where the video stream generated by the *DVS* is actually displayed. An *MPEG Decoder* module receives the MPEG TS video stream distributed by the remote *Streaming Service*, it extracts synchronization data, decodes MPEG frames, and displays ready images according to received timing information. Video processing capabilities depend on a general purpose open-source multimedia library derived from the VLC player (Videolan, 2006). An example showing the usage of the proposed architecture is given in Figure 5.

3-D Visualization on Mobile Devices*Figure 4. Layout of proposed streaming-based remote rendering architecture**Figure 5. a) Remote visualization session involving a PDA and a visualization console located at the DVS site. b) Detail of user interaction with the remote 3-D scene.***Experimental Results**

The effectiveness of the proposed architecture has been evaluated in remote visualization scenarios where mobile users equipped with PDA and Tablet PC devices simultaneously interact with a general

purpose 3-D surface rendering application running on an accelerated graphics back-end. The main aim is to prove that such a remote visualization framework allows for highly interactive collaborative sharing of realistic virtual worlds, between a high-end visualization console and multiple

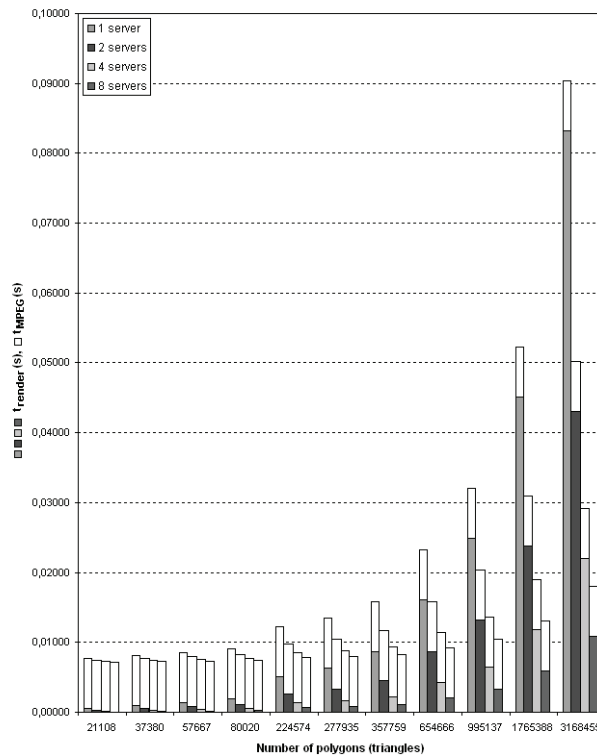
3-D Visualization on Mobile Devices

mobile users, regardless of the complexity of the scene being considered. Since visualization frame rate and overall latency experienced at the mobile client both constitute the main limitations of existing remote rendering architectures, we provided a quantitative analysis of the critical parameters, thus giving a precise measure of the achieved degree of interactivity. In particular, the time needed to render a single image (t_{render}), to generate the corresponding encoded frame (t_{MPEG}) and to display it on the mobile device ($t_{streaming}$) has been measured. Other important parameters include the time needed for a command to reach the *DVS* ($t_{command}$), the frame rate at the *DVS* (fps_{DVS}), the frame rate at the mobile device (fps_{client}), and the average bitrate ($bitrate_{client}$).

Experiments have been carried out on a chromium-based rendering subsystem including up to eight nodes, each running RedHat Linux Fedora 3 and equipped with a Pentium IV 2

GHz, an nVidia Quadro FX-1100 AGP graphics accelerator with 128-MB video memory, 256-MB main memory, and a GB Ethernet network card. We demonstrated that, for a given visualization resolution, interactive scene manipulation can be experienced by simply removing the bottleneck given by the frame rate at the *DVS*, assuming that the available bandwidth is capable of supporting video streaming requirements. Measurements for a 240×240 rendering resolution and streaming video size and for increasing scene complexity and rendering nodes are illustrated in Figure 6. Corresponding frame rate at the *DVS* is reported in Figure 7. Since the *Mobile Viewer* is able to manage MPEG visualization at up to 30 fps, frame rate at the client side is not reported since it always corresponds to fps_{DVS} when fps_{DVS} is lower than 30 fps, and to approximately 30 frames per second when fps_{DVS} is larger than 30 fps. It can be observed that for high complexity scenes, t_{render}

Figure 6. Performance with a rendering resolution and streaming video size of 240×240 pixels for increasing number of polygons and video servers



3-D Visualization on Mobile Devices

Figure 7. Frame rate with a rendering resolution and streaming video size of 240×240 pixels for increasing number of polygons and video servers.

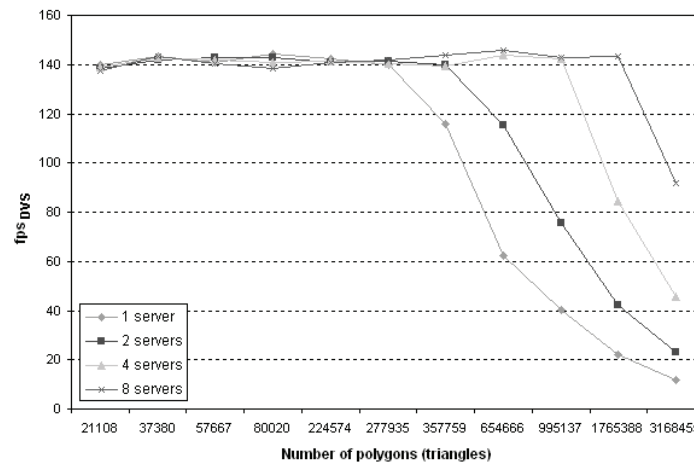
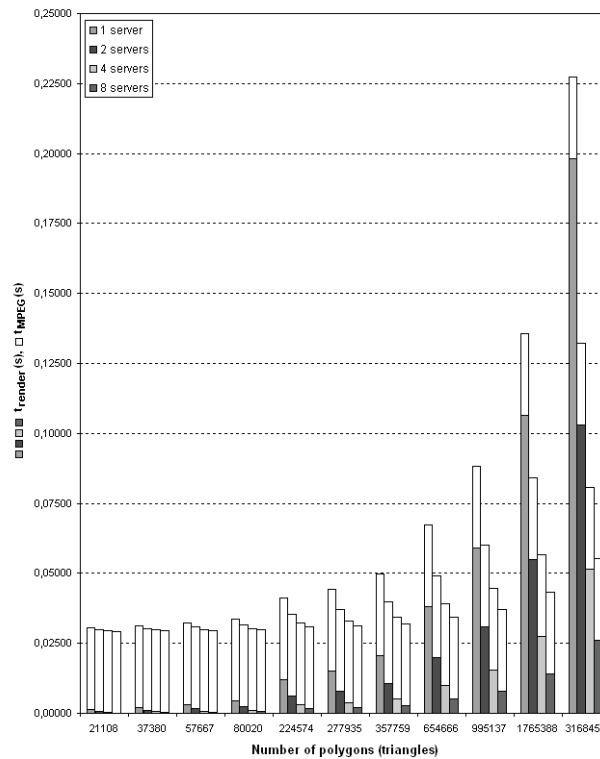


Figure 8. Performance with a rendering resolution and streaming video size of 512×512 pixels for increasing number of polygons and video servers.



3-D Visualization on Mobile Devices

becomes predominant over t_{MPEG} and the degree of interactivity is greatly reduced. Nevertheless, by increasing the number of rendering servers, t_{render} can be reduced, and interactive frame rates can be experienced. It is worth observing that all the considered scenes can be visualized at the maximum frame rate using only four servers. Results for a rendering resolution and streaming video size of 512×512 pixels are reported in Figures 8 and 9. In this case, without introducing a rendering distribution layer, even scenes with less than one million polygons can be navigated only with a low frame rate. However, using eight rendering servers, all the scenes can be visualized on the mobile device at 30 fps.

To conclude the evaluation of system behavior, several experiments were carried out to assess the impact of the main parameters of the *Video SPU* and *Encoding Service* as well as of the communication link being used on the overall latency between scene manipulation and display update. For this, four Chromium server nodes were used to allow the navigation of a high complexity 3-D scene (one million polygons per frame) at different rendering resolutions and visualization sizes ranging from 240×240 to 1024×1024 pixels. Moreover, different bitrates were used for the MPEG stream. Experiments were carried out in various wireless networking environments including GPRS, UMTS

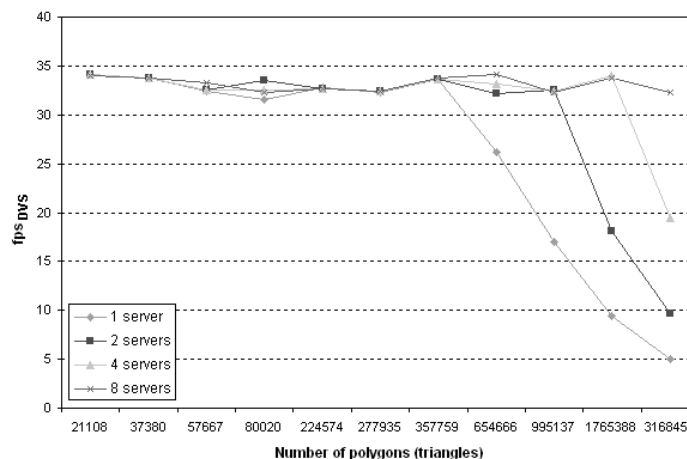
and WLAN IEEE 802.11b. Overall latency for all the considered configurations is reported in Figure 10, where the contributions of $t_{command}$, t_{render} , t_{MPEG} , and $t_{streaming}$ and the corresponding frame rate are accurately quantified.

According to experimental results, we can conclude that latency in WLAN environments allows for very interactive scene manipulation, with an average response time of 0.42 s. Latency measured in GPRS- and UMTS-based communication scenarios can be possibly unacceptable when dealing with applications requiring real-time like interaction. Nevertheless, they can certainly ensure reasonable response times (around 0.83 s) in all those professional contexts where the goal is to enable visualization of highly complex 3-D scenes over large geographical distances.

CONCLUSION AND FUTURE TRENDS

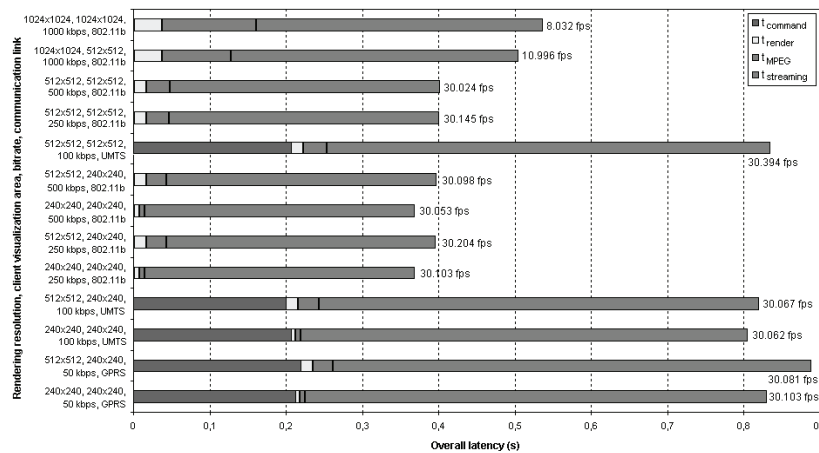
In this chapter, a review of the main solutions for developing 3-D visualization applications on mobile devices is presented. Local computation-based techniques simply exploiting rendering resources available on the mobile side are considered first. Here, graphics support can be provided by general purpose CPUs or by specific GPUs. In the first case,

Figure 9. Frame rate with a rendering resolution and streaming video size of 512×512 pixels for increasing number of polygons and video servers.



3-D Visualization on Mobile Devices

Figure 10. Impact of rendering resolution, visualization area size, video stream bitrate and communication channel on the degree of usability experienced by the mobile user in terms of overall latency and frame rate



only limited complexity scenes can be rendered at interactive frame rates. On the other hand, when specialized hardware is available, complex scenes can be smoothly navigated. Nevertheless, manipulation of extremely complex scenes required in specialized applications like remote diagnosis, collaborative CAD/CAM sessions, and scientific visualization becomes possible only through remote computation/visualization techniques. The application of remote visualization to the mobile world is relatively new, since obstacles to its diffusion have been overcome only in recent years, thanks to the advances in multimedia and communication technologies. Nevertheless, a solution allowing local-like manipulation of highly complex 3-D worlds on hand-held devices by means of distributed rendering and streaming techniques is presented in this chapter, and aspects related to the degree of interactivity perceived by mobile users are investigated in depth.

As regards future research activities in the field of mobile 3-D, it is worth observing that a crucial role will be certainly played by hand-held GUIs, 3-D APIs for mobile and embedded graphics accelerators. Next-generation GUIs will have to be able to provide effective interaction capabilities. In fact, one of the most limiting factors to the diffusion of

mobile 3-D is related to the poor human-computer interfaces available on mobile terminals, including limited display and constrained input devices. Thus, it will have to be decided whether to present the mobile user with the same GUI available at the remote site, to recreate a simplified version of the interface or to design specific interfaces specifically tackling actual device constraints. From the software point of view, the future trend seems to be very similar to the scenario already seen for desktop PCs, where OpenGL and Direct3D represent the main technologies for developing 3-D graphics applications. Today, OpenGL ES and Direct3D Mobile are candidates for dominating the market of APIs for mobile systems. The trend for mobile hardware is also clearly outlined: all PDAs, smartphones, and other mobile devices will soon be equipped with specialized hardware for 2-D and 3-D graphics. Thus, local visualization will become increasingly attractive and feasible. However, remote visualization techniques, like the one presented in this chapter, will continue to play a key role in all those scenarios in which the complexity of the scene exceeds the hardware capabilities of the client device, and in which only data representations can be delivered over the network because of security constraints.

3-D Visualization on Mobile Devices**ACKNOWLEDGMENT**

This chapter is part of a work developed within the European Research Project VICTORY, “Audio-Visual Content search and retrieval in a distributed P2P repository.” VICTORY is funded under the Information Society Technologies (IST) thematic program of the European Commission’s Sixth Framework Program.

REFERENCES

- Beerman, D. (2003, October). *CRUT: Event distribution and image delivery for cluster-based remote visualization*. Paper presented at IEEE Visualization 2003—Workshop on Parallel Vis. and Graphics, Seattle, Washington.
- Burigat, S., & Chittaro, L. (2005). Location-aware visualization of VRML models in GPS-based mobile guides. In *Proceedings of the 10th International Conference on 3D Web Technology* (pp. 57-64).
- Chang, C. F., & Ger, S. H. (2002). Enhancing 3D graphics on mobile devices by image-based rendering. In *Proceedings IEEE Third Pacific-Rim Conference on Multimedia* (pp. 1105-1111).
- Engel, K., Ertl, T., Hastreiter, P., Tomandl, B., & Eberhardt, K. (2000). Combining local and remote visualization techniques for interactive volume rendering in medical applications. *IEEE Visualization*, 449-452.
- Engel, K., Sommer, O., & Ertl, T. (2000). A framework for interactive hardware accelerated remote 3D-visualization. In *Proceedings of the Eurographics—IEEE TCVG Symposium on Visualization* (pp. 167-177).
- Garland, M. (1999). Multiresolution modeling: Survey & future opportunities. In *Eurographics '99 State of the Art Reports* (pp. 111-131).
- Glidden, R. (1997). *Graphics programming with Direct3D*. Addison Wesley Longman.
- Grimstead, I. J., Avis, N. J., & Walker, D. W. (2005). Visualization across the pond: How a wireless PDA can collaborate with million-polygon datasets via 9,000km of Cable. In *Proceedings of the ACM/SIGGRAPH Web3D Symposium* (pp. 47-56).
- Humphreys, G., Houston, M., Ng, R., Frank, R., Ahern, S., Kirchner, P., & Klosowski, J. T. (2002). Chromium: A stream-processing framework for interactive rendering on clusters. In *SIGGRAPH 02 Conference Proceedings* (pp. 693-702).
- Hybrid’s Gerbera Website. (2006). Retrieved June 12, 2006, from <http://www.hybrid.fi/>
- IBM, Deep Computing Visualization Website. (2006). Retrieved June 9, 2006 <http://www-03.ibm.com/servers/deepcomputing/>
- Intel Xscale technology specifications. (2006). Retrieved June 12, 2006, from <http://www.intel.com/design/intelxscale/>
- Klimt Website. (2006). Retrieved June 12, 2006, from <http://studierstube.org/klimt/>
- Lamberti, F., Zunino, C., Sanna, A., Fiume, A., & Maniezzo, M. (2003). An accelerated remote graphics architecture for PDAs. In *Proceedings of the ACM/SIGGRAPH Web3D Symposium* (pp. 55-61).
- Luebke, D.P. (2001). A developer’s survey of polygonal simplification algorithms. *IEEE Computer Graphics and Applications*, 21(3), 24-35
- M3D Website. (2006). Retrieved June 12, 2006, from <http://www.jcp.org/en/jsr/>
- M3G - JSR 184 Web site. (2006). Retrieved June 12, 2006, from <http://www.hybrid.fi/main/m3g/>
- Martin, I. M. (2000). *Adaptive rendering of 3D models over networks using multiple modalities* (Tech. Rep.). IBM T.J. Watson Research Center.
- McMillan, L. (1999). Image-based rendering: A new interface between computer vision and computer graphics. *SIGGRAPH Computer Graphics Newsletter—Applications of Computer Vision to Computer Graphics*, 33(4), 61-64.
- McMillan, L., & Bishop, G. (1995). Plenoptic modeling: An image-based rendering system.

3-D Visualization on Mobile Devices

In *SIGGRAPH 95 Conference Proceedings* (pp. 39–46).

Mental images RealityServer Web site. (2006). Retrieved June 12, 2006, http://www.mental-images.com/2_3_realityserver/

Mental images (2005). *RealityServer Functional Overview*. (Tech. Rep. No. 005-01). Retrieved June 5, 2006, from <http://www.mentalimages.com/cgi-bin/rs-whitepaper.cgi>

Microsoft Direct3D Mobile (2006). Retrieved January 2, 2007, from http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wcemultimedia5/html/wce50_oriDirect3DMobile.asp

MiniGL Website. (2006). Retrieved June 12, 2006, from <http://www.dsbox.com/minigl.html>

MPEG1 - ISO/IEC 11172. (1992). *Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s*.

MPEG2 - ISO/IEC 13818. (1994). *Generic coding of moving pictures and associated audio information*.

OpenGL ES Web site. (2006). Retrieved June 12, 2006, from <http://www.khronos.org/opengles/>

Parallel Graphics Web site. (2006). Retrieved June 12, 2006, from <http://www.parallelgraphics.com/>

PocketGL Web site. (2006). Retrieved June 12, 2006, from <http://www.pocketgear.com>

Richardson, T., Stafford-Fraser, Q., Wood, K. R., & Hopper, A. (1998). Virtual network computing. *IEEE Internet Computing*, 2(1), 33–38.

Silicon Graphics, Inc. OpenGL Vizserver Web site (2005). Retrieved December 22, 2005, from <http://www.sgi.com/software/vizserver/>

Stegmaier, S., Diepstraten, J., Weiler, M., & Ertl, T. (2003). Widening the remote visualization bottleneck. In *Proceedings of the 3rd International Symp. on Image and Signal Proc. and Analysis* (pp. 174-179).

Stegmaier, S., Magallón, M., & Ertl, T. (2002). A generic solution for hardware-accelerated remote visualization. In *Proceedings of Eurographics—IEEE TCVG Symp. on Visualization* (pp. 87-94).

Videolan project Web site. (2006). Retrieved June 12, 2006, from <http://www.videolan.org>

Vincent Web site. (2006). Retrieved June 12, 2006, from <http://ogl-es.sourceforge.net/>

VirtuaGL project Web site (2006). Retrieved June 12, 2006, from <http://virtualgl.sourceforge.net/>

Woo, M., Neider, J., Davis, T., & Shreiner, D. (1999). *OpenGL programming guide: The official guide to learning OpenGL—ver. 1.2* (3rd ed.). Addison-Wesley.

Woodward, C., Valli, S., Honkamaa, P., & Hakkarainen, M. (2002). Wireless 3D CAD viewing on a PDA device. In *Proceedings of the 2nd Asian Mobile Computing Conference* (pp. 83-87).

Zhou, H. (2006). *A survey on ubiquitous graphics* (Tech. Rep.). Hong Kong University of Science and Technology. Retrieved December 18, 2006, from <http://www.cse.ust.hk/~zhouhong/publications.html/>

Zunino, C., Lamberti, F., & Sanna, A. (2003). A 3-D Multiresolution rendering engine for PDA devices, In *SCI 2003 Proceedings, vol. V* (pp. 538-542).

KEY TERMS

API: An application programming interface (API) is a language and message format used by an application program to communicate with the operating system or some other control program such as a database management system or communications protocol.

GPU: Graphics processing units are specialized logic chips devoted to rendering 2-D or 3-D images. Display adapters contain one or more GPUs for fast graphics rendering.

3-D Visualization on Mobile Devices

GUI: A graphics user interface (GUI) is a particular case of user interface for interacting with a computer that employs graphic images and widgets in addition to text to represent the information and actions available to the user. Usually, actions are performed through direct manipulation of the graphic elements.

Local Visualization: In local visualization (or local computation) graphics tasks are carried out by exploiting the capabilities of the actual device.

Mobile/Portable/Hand-Held Devices: PDA, Tablet PC, Smartphone: Hand-held devices are pocket-sized computing devices, typically using a small display for user output and a miniaturized keyboard for user input. Input and output devices can be combined into a single touch-screen based interface. These devices are becoming increasingly popular amongst those users requiring the assistance and convenience of a conventional computer in a mobile and wireless environment.

Multimodal Interfaces: Humans perceive the world through senses; communication through one sense is known as a mode. Multimodal interfaces try to combine two (or more) modes of communication to enhance user interaction.

Polygon: A polygon is a simple closed curve consisting entirely of a finite number of line segments. A polygon mesh is a 3-D object comprising a certain number of polygons joined at their edges and vertices.

Remote Visualization: In remote visualization (or remote computation) heavy graphics tasks are delegated to a high-end graphics server that actually performs the 3-D rendering and generates a 2-D frame that can be visualized on a remote device (possibly characterized by limited hardware resources).

Rendering: Creating an image representing objects designed in a three-dimensional modeling program.

Streaming Video: A one-way video transmission over a data network. It is widely used on the Web as well as on private intranets to deliver video-on-demand. Unlike movie files that are played after they are downloaded, streaming video is played within a few seconds of requesting it, and the data is not stored permanently in the computer.

3-D Computer Graphics: 3-D computer graphics are works of graphic art that are created with the aid of digital computers and specialized 3-D software.

Voxel: A voxel (volume pixel) represents a quantity of 3-D data just as a pixel represents a point or a cluster of points in 2-D data. It is used in scientific and medical applications that process 3-D images.

InfoSci®-Books

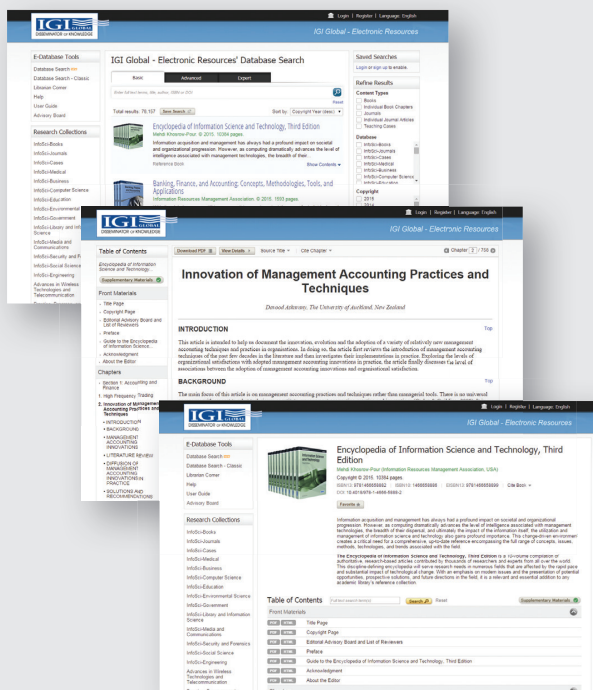
A Database for Progressive Information Science and Technology Research

Maximize Your Library's Book Collection!

Invest in IGI Global's InfoSci®-Books database and gain access to hundreds of reference books at a fraction of their individual list price.

The InfoSci®-Books database offers unlimited simultaneous users the ability to precisely return search results through more than 68,000 full-text chapters from nearly 3,000 reference books in the following academic research areas:

Business & Management Information Science & Technology • Computer Science & Information Technology
Educational Science & Technology • Engineering Science & Technology • Environmental Science & Technology
Government Science & Technology • Library Information Science & Technology • Media & Communication Science & Technology
Medical, Healthcare & Life Science & Technology • Security & Forensic Science & Technology • Social Sciences & Online Behavior



Peer-Reviewed Content:

- Cutting-edge research
- No embargoes
- Scholarly and professional
- Interdisciplinary

Award-Winning Platform:

- Unlimited simultaneous users
- Full-text in XML and PDF
- Advanced search engine
- No DRM

Librarian-Friendly:

- Free MARC records
- Discovery services
- COUNTER4/SUSHI compliant
- Training available

To find out more or request a free trial, visit:
www.igi-global.com/eresources

